TITLE PAGE

Lewis University  
CPSC 50900: Database Systems  
Term Project

TITLE OF YOUR PROJECT: MUSIC STORE DATABASE SYSTEM

First Person: Pasha Mohammed

First Person Email Address: pashamohammed@lewisu.edu

Second Person: Manideep Addagudi

Second Person Email Address: manideepaddagudi@lewisu.edu

Third Person: BHANU Prakash MASHETTY

Third Person Email Address: bhanuprakashmashet@lewisu.edu

Work products stored in the Github repository: https://github.com/Pmohammed12/DATABASE-SYSTEMS-PROJECT-003.git

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

*Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?*

*Rubric: Your response to each of these six questions will be graded out of 3 points.*

* *3 points: clear, complete descriptions that convey the importance and meaning of your data*
* *2 points: mostly clear descriptions, although some additional data would have helped in some sections*
* *1 point: necessary details are lacking in many of your responses.*

*You will also earn 2 additional points for coming up with a descriptive title for your project.*

*Total points possible: 20*

ENTER YOUR INITIAL PROPOSAL HERE

This is to develop a database for a music store. The database will contain information about

1. The admin details; here the information about the person who will administrate the system, s/he will perform all database activities on the system
2. Music inventory; this the music store i.e. music genre, size, artist and name
3. Viewers details; the customer information is given here; the customer will login to the system and key in the required data to access the music store.
4. Sales; the sale details monitored and stored in the database.
5. Views; the users who accessed and played music from the inventory

The interest on music is derived from the urge of the rise of demands of the entertainment sector

These data is important to make a successful system that can manage the store and enable viewers to access and play the music from the store.

Data will be obtained from various sources; the admin will be the one responsible for availing the data in the music inventory, online and viewers.

The data will be utilized by the music lovers who want to entertain themselves with the music from the store.

A web based application

# Data Sources

*Description:* *Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or some other custom format. Not all the files need be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it structured, and describing how you will reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.*

*Rubric: Your work will be graded as follows:*

* *5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.*
* *5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.*
* *3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.*
* *2 points: you post the data files to your GitHub account and make it possible for me to see them.*

*Total points possible: 15*

ENTER YOUR DATA SOURCES DESCRIPTION HERE

The main data source is the database. The xampp is used as the local sever. The tables inside the



This the .csv data source from database

It has six tables in the database the inventory table for storing music, register table that enables the new users to create account with the system to access the music store,artist which store information about all musicians using the store ,music\_genre which explains the various types of songs that are stored, .

In the table.1 (music\_store) there five fields:

Music\_id name of the field, int- the data type; the primary key and also indicate the number of songs in the database

Music\_artist- field name, varchar- the data type; the name of the artist who composed the song

Music\_name - field name, varchar- the data type; the name of the music in the inventory

Music\_genre- field name, varchar- the data type; the type of the music in the inventory

Music\_year- field name, varchar- the data type; the year the of the music first production

In the table.2 (register) there five fields:

User\_id- field name, varchar- data type; the primary key and indicate the number of users stored in the database

Fullname- field name, varchar-data type; takes the full names of the user

Username- field name, varchar-data type; takes the name that will be used to login into the system by the user

Email- field name, varchar-data type; the email will be used by the admin to login into the system

Password- field name, varchar-data type; the login credentials to the system

In the table.3 (artist) there five fields:

| **artist\_id**- field name, varchar- data type; the primary key and indicate the number of users stored in the database |  |  |  |  |
| --- | --- | --- | --- | --- |

**FULLNAMES** - field name, varchar-data type; takes the full names of the user

**NICKNAME** - field name, varchar-data type; takes the name that will be used to login into the system by the user

**PRODUCTION\_LABEL** - field name, varchar-data type; the email will be used by the admin to login into the system

**MUSIC\_GENRE** - field name, varchar-data type; the login credentials to the system

In the table.3 (register) there five fields:

| **artist\_id**- field name, varchar- data type; the primary key and indicate the number of artist using the system |  |  |  |  |
| --- | --- | --- | --- | --- |

**FULLNAMES** - field name, varchar-data type; takes the full names of the user

**NICKNAME** - field name, varchar-data type; the other name that the artist is popularly known with other than the official names

**PRODUCTION\_LABEL** - field name, varchar-data type; the label that the artist is signed to

**MUSIC\_GENRE** - field name, varchar-data type; the type of music that the artist sings

In the table.4 (tracks) there three fields:

track\_Id-name of the field, int- the data type; the primary key

Track\_number- field name, int- the data type; the exact number label of the track in the store

Track\_name- field name, varchar- the data type; the name of the track

Music\_id- field name, int- the data type; the foreign key linking table tracks and table music store

Track\_url field name , vachar- the location of the track in the database

In the table.5 (record\_label) there four fields:

label\_Id-name of the field, int- the data type; the primary key

| **LEBAL\_NAME**- field name, varchar- the data type; the name of the label |  |  |
| --- | --- | --- |

**MUSIC\_GENRE** - field name, varchar- the data type; the specific genre of the music

**SERVICES**- field name, varchar- the data type; the services that the label offers

In the table.6 (views) there three fields:

view\_Id -name of the field, int- the data type; the primary key

| **VIEWER\_NAME**- field name, varchar- the data type; the name of the viewer |  |  |
| --- | --- | --- |

**DATE\_VIEWED** - field name, varchar- the data type; the date that the view visited the system

The table will be linked to a form where users of the system will key in some data into the database

The system also will have computer file which will have several pieces of code to make the project work as expected. There other data sources for the project i.e. php file, database connection file , html files, css files etc. all this will be combined and programmed together to present a well working database system.

“The data need to be posted on your Github account”

# Data Storage Alternatives

*Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.*

*Rubric: Your work will be graded as follows*

* *5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.*
* *5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.*

*Total points possible: 10*

ENTER YOUR ALTERNATIVE DATA STORAGE IDEAS HERE

IN-MEMORY DATABASE

An in-memory database (IMDB) stores computer data in a computer’s main memory instead of a disk drive to produce quicker response times. Accessing data stored in memory eliminates the time needed to query data from a disk. In-memory databases are used by applications that depend on rapid response times and real-time data management. Industries that benefit from in-memory databases include telecommunications, banking, travel and gaming. An in-memory database is also referred to as a main memory database (MMDB), real-time database (RTDB) or in-memory database system (IMDS).

IMDB keeps all its data in random access memory (RAM) of a computer. Main memory is only accessed when querying data

ADVANTAGES OF IN-MEMORY DATABASE

* Faster transactions
* No translation
* Multi-user concurrency

DISADVANTAGES OF IN-MEMORY DATABASE

* The data stored is only temporary
* Cannot store large amount of data
* Replication of data files
* The collection of data from a growing number of sources

NoSQL DATABASE

NoSQL database is **an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.** In that data is not stored using fixed table schemas. Mainly its purpose is **to serve as database system for huge web-scale applications** (Amazon, Google, Facebook, etc...)

**ADVANTAGES OF NOSQL DATABASES**

* Handle large volumes of data at high speed with a scale-out architecture.
* Store unstructured, semi-structured, or structured data.
* Enable easy updates to schemas and fields.
* Be developer-friendly.
* Take full advantage of the cloud to deliver zero downtime.

**DISADVANTAGES OF NOSQL DATABASES**

* Not all NoSQL databases contemplate the atomicity of instructions and the integrity of the data. ...
* Compatibility issues with SQL instructions. ...
* Lack of standardizing. ...
* Cross-platform support. ...
* They usually have not-really-useful management tools or console access.

# Relational Database Design Process

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE SOURCE CODE AND SCREEN SHOTS.

Functional dependency (FD) determines the relation of one attribute to another in DBMS. FD aids to maintain the quality of data in the database. Functional dependency of X on Y is represented by FD= X→Y

* 1. in table artist, there 5 attribute: artist\_id uniquely identifiers all other attributes i.e. **fullnames, nickname, production\_label and music\_genre**

**artist\_id (determinant) →** **fullnames, nickname, production\_label and music\_genre** (dependent)

* 1. in table tracks, there 5 attributes : **track\_id uniquely identifiers the four other attribute i.e. track\_number, track\_name, music\_id and track\_url**

**track\_id (determinant) →track\_number, track\_name, music\_id and track\_url** (dependent)

* 1. in table record\_lebal there 4 attribute: lebal\_id uniquely identifies the **lebal\_name,**  **music\_genre and services**
  2. in table register, there 5 attribute: user\_id uniquely identifies the **fullname, username, email and password**

**label\_id(determinant)→ fullname, username, email and password**(dependent)

* 1. **In table music\_store, there five attribute: music\_id uniquely identify music\_name, music\_artist, music\_year and music\_url.**

**Music\_id (determinant)→music\_name, music\_artist, music\_year and music\_url**

**(dependent)**

* 1. in table views, there 3 attribute: view\_id uniquely identify the viewer\_name and date\_viewed

view\_id**(determinant)→** viewer\_name and date\_viewed **(dependent)**

ENTITIES

* Artist
* Music store
* Production\_Label
* Track
* Register
* Views

|  |
| --- |
| artist |
| (pk)Artist\_id  Fullnames  Nickname  Production\_label  Music\_genre |

|  |
| --- |
| record\_label |
| (pk)label\_id  **label\_name**  **music\_genre**  **services** |

|  |
| --- |
| Music\_store |
| (pk)music\_id  **music\_name**  **music\_artist music\_id music\_year** |

|  |
| --- |
| tracks |
| (pk)Track\_id  **track\_number**  **track\_name music\_id track\_url** |

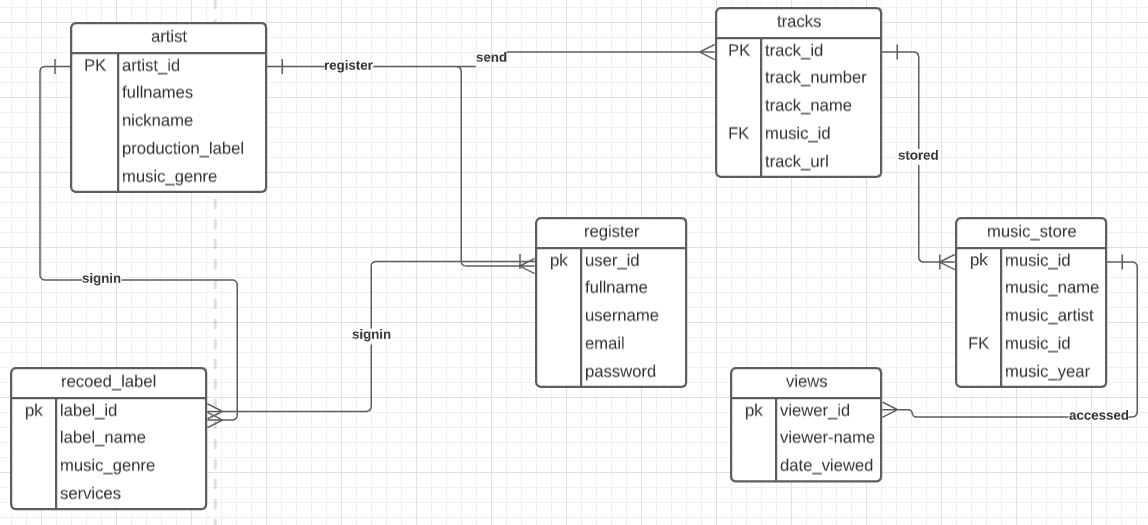
entity

attributes

|  |
| --- |
| views |
| (pk)view\_id  **viewer\_name**  **view\_date** |

|  |
| --- |
| register |
| (pk)user\_id  fullname  **username**  **email**  **password** |

ER DIAGRAM



# Figure 1.1

# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

ENTER YOUR RELATIONAL DATABASE DESIGN HERE

The database obeys the first normal form rule since it uses scalable table design which can be easily extended and data can be retrieved from the database by using various sql queries.

The tables in the database music\_store follows the rules of 1st normal form.

* The column contain atomic values e.g. in table music\_store we have columns music\_id, music\_name etc
* The column should contain values of the same type; date \_viewed should contain only data of date format
* Each column should have a unique name e.g. track\_name, artist\_name etc.
* Order in which data is saved does not matter

2nd normal form

* The tables have qualified to be in the 1st normal form which is requirement in 2nd normal form.
* The tables have no partial dependencies as we show in the relational database design process.
* Where every table has a unique primary key that can extract any data from the tables even when other attributes contain like data

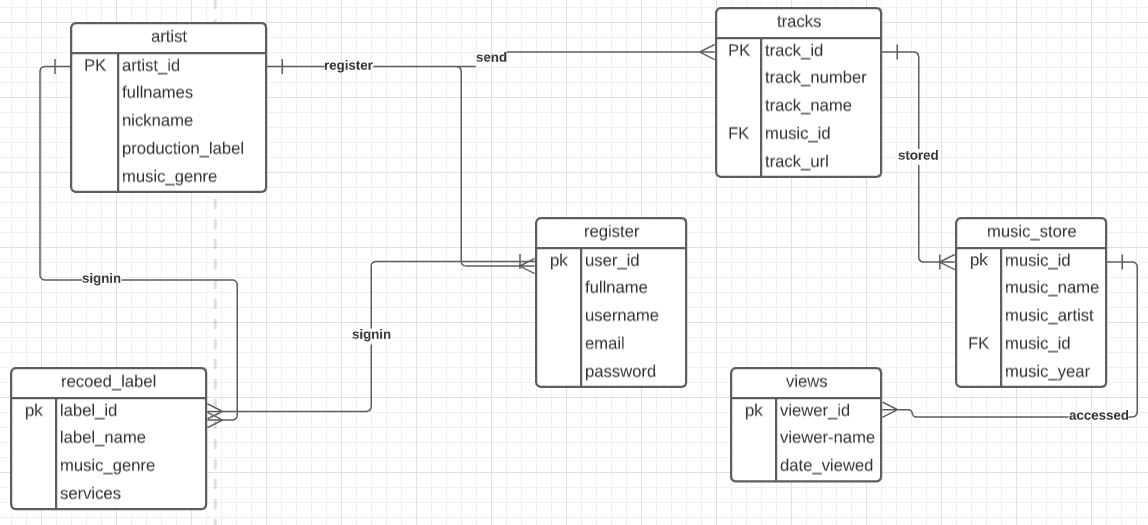
3rd normal form

* Should be in 2nd normal form
* It’s should not contain have transitive dependencies. In our data we have done away with non-primary key columns dependency whereby all dependent relies on the primary key only.

4th normal form

* Should be in 3rd normal form
* No multivalued dependency should exist

Bridge relationship



**database physical model for musicStore**

Database model documentation

Created with Vertabelo.com

# 1. Model details

**Model name:**database physical model for musicStore

**Version:**2.4

**Database engine:**MySQL

# 2. Tables

## 2.1. Table register

**Description:**enables the user of the system to signup

2.1.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| user\_id | int | PK |  |
| fullname | varchar |  |  |
| username | varchar |  |  |
| email | varchar |  |  |
| password | varchar |  |  |
| views\_viewer\_id | int |  |  |

## 2.2. Table music\_store

2.2.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| music\_id | int | PK |  |
| music\_name | varchar |  |  |
| music\_artist | varchar |  |  |
| music\_genre | varchar |  |  |
| music\_year | date |  |  |
| artist\_artist\_id | int |  |  |
| record\_label\_label\_id | int |  |  |
| tracks\_track\_id | int |  |  |

## 2.3. Table record\_label

**Description:**highlights the label available for artists and services they offer

2.3.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| label\_id | int | PK |  |
| label\_name | varchar |  |  |
| music\_genre | varchar |  | highlight the type of music that the specific record label deals with majorly to the artist |
| services | varchar |  |  |

## 2.4. Table tracks

**Description:**the music list

2.4.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| track\_id | int | PK |  |
| track\_number | int |  |  |
| track\_name | varchar |  |  |
| music\_id | int |  |  |
| track\_url | varchar |  |  |

## 2.5. Table views

**Description:**the table shows the persons who have accessed the database

2.5.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| viewer\_id | int | PK |  |
| viewer\_name | VARCHAR |  |  |
| date\_viewed | date |  |  |

## 2.6. Table artist

**Description:**offers artists details

2.6.1. Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Type** | **Properties** | **Description** |
| artist\_id | int | PK |  |
| fullnames | varchar |  |  |
| production\_label | varchar |  |  |
| music\_genre | varchar |  |  |
| register\_user\_id | int |  |  |

# 3. References

## 3.1. Reference register\_views

|  |  |  |
| --- | --- | --- |
| **views** | **1..\*** | **register** |
| viewer\_id | <-> | views\_viewer\_id |

## 3.2. Reference music\_store\_artist

|  |  |  |
| --- | --- | --- |
| **artist** | **0..\*** | **music\_store** |
| artist\_id | <-> | artist\_artist\_id |

## 3.3. Reference music\_store\_record\_label

|  |  |  |
| --- | --- | --- |
| **record\_label** | **0..\*** | **music\_store** |
| label\_id | <-> | record\_label\_label\_id |

## 3.4. Reference artist\_register

|  |  |  |
| --- | --- | --- |
| **register** | **0..\*** | **artist** |
| user\_id | <-> | register\_user\_id |

## 3.5. Reference music\_store\_tracks

|  |  |  |
| --- | --- | --- |
| **tracks** | **0..\*** | **music\_store** |
| track\_id | <-> | tracks\_track\_id |

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be grades as follows:*

* *Database and table creation statements from Vertabello saved as an sql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

ENTER YOUR DDL WORK HERE

VERTABELO SQL SCRIPT



SCRIPTS USED TO POPULATE THE TABLES IN THE DATABASE

* + INSERT INTO `views` (`VIEW\_ID`, `VIEWER\_NAME`, `DATE\_VIEWED`) VALUES ('03', 'almado kigali', '2021-11-01');
  + INSERT INTO `tracks` (`track\_id`, `track\_number`, `track\_name`, `music\_id`, `track\_url`) VALUES ('4', '14', 'enjoyment', NULL, 'This PC\\TECNO W5 Lite\\Internal storage\\VidMate\\download');
  + INSERT INTO `register` (`USER\_ID`, `FULLNAME`, `USERNAME`, `EMAIL`, `PASSWORD`) VALUES (NULL, ' kamutu gibson', 'kamatu', 'kamatugibson@gmail.com', '654321');
  + INSERT INTO `record\_lebal` (`LEBAL\_ID`, `LEBAL\_NAME`, `music\_genre`, `SERVICES`) VALUES ('4', 'wasafi records', 'bongo', 'production and promotion');
  + INSERT INTO `music\_store` (`music\_id`, `music\_name`, `music\_artist`, `music\_genre`, `music\_year`) VALUES ('5', 'enjoyment', 'kidi', 'dancehall', '2020'
  + INSERT INTO `artist` (`artist\_id`, `FULLNAMES`, `NICKNAME`, `PRODUCTION\_LABEL`, `MUSIC\_GENRE`) VALUES ('4', 'diamond plantinuz', 'simba', 'wasafi records', 'bongo');

Description of script;

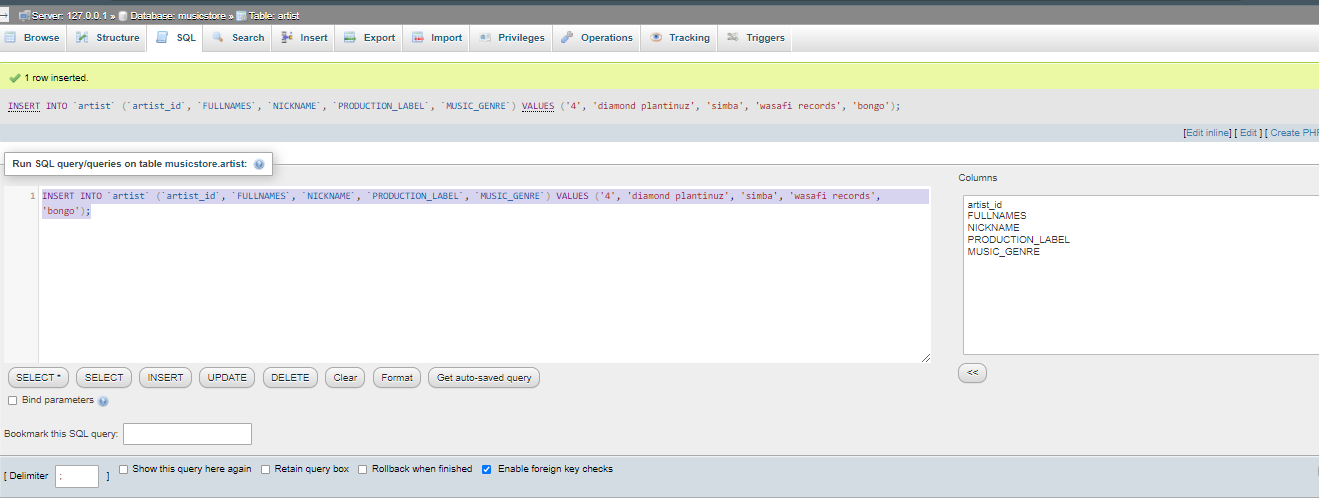
* + INSERT INTO `views` (`VIEW\_ID`, `VIEWER\_NAME`, `DATE\_VIEWED`) VALUES ('03', 'almado kigali', '2021-11-01'); → this populate the table views by insert a row of data.
  + INSERT INTO `tracks` (`track\_id`, `track\_number`, `track\_name`, `music\_id`, `track\_url`) VALUES ('4', '14', 'enjoyment', NULL, 'This PC\\TECNO W5 Lite\\Internal storage\\VidMate\\download');→ this populate the table tracks by insert a row of data.
  + INSERT INTO `register` (`USER\_ID`, `FULLNAME`, `USERNAME`, `EMAIL`, `PASSWORD`) VALUES (NULL, ' kamutu gibson', 'kamatu', 'kamatugibson@gmail.com', '654321');→ this populate the table register by insert a row of data.

→

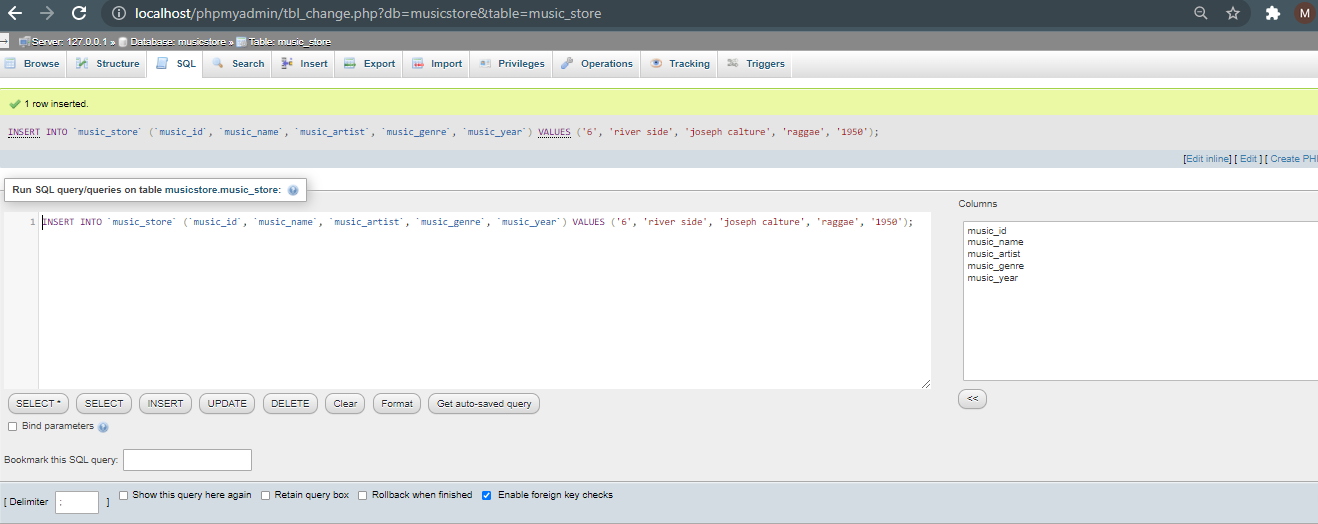
* + INSERT INTO `record\_lebal` (`LEBAL\_ID`, `LEBAL\_NAME`, `music\_genre`, `SERVICES`) VALUES ('4', 'wasafi records', 'bongo', 'production and promotion');→ this populate the table record\_label by insert a row of data.
  + INSERT INTO `music\_store` (`music\_id`, `music\_name`, `music\_artist`, `music\_genre`, `music\_year`) VALUES ('5', 'enjoyment', 'kidi', 'dancehall', '2020');→ this populate the table music\_store by insert a row of data.
  + INSERT INTO `artist` (`artist\_id`, `FULLNAMES`, `NICKNAME`, `PRODUCTION\_LABEL`, `MUSIC\_GENRE`) VALUES ('4', 'diamond plantinuz', 'simba', 'wasafi records', 'bongo');→ this populate the table artist by insert a row of data.

Successful screenshot:

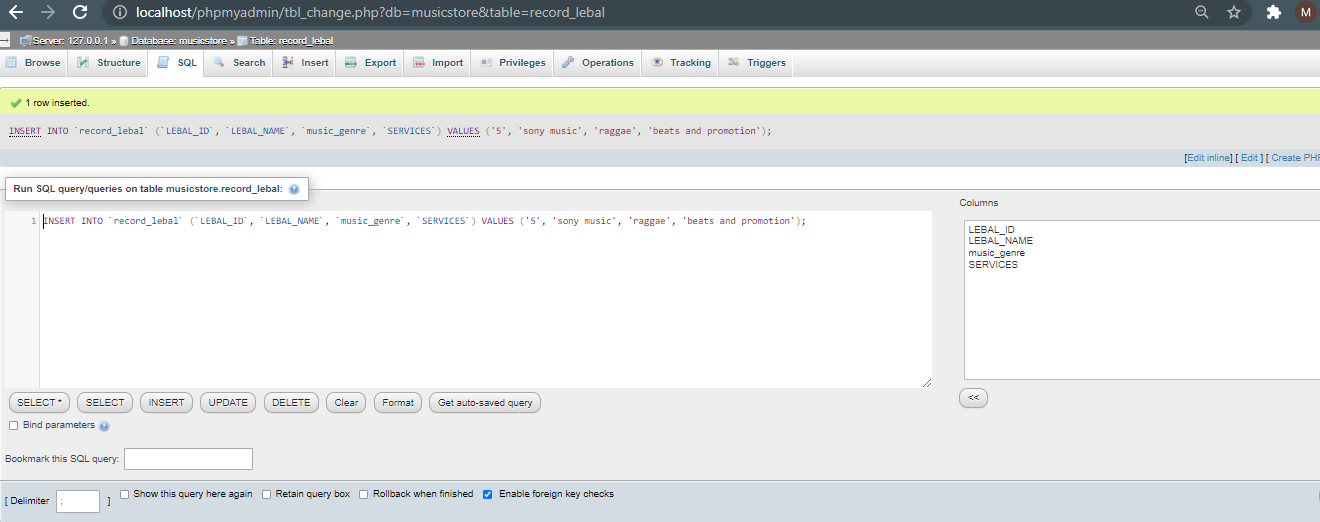
In table artists;



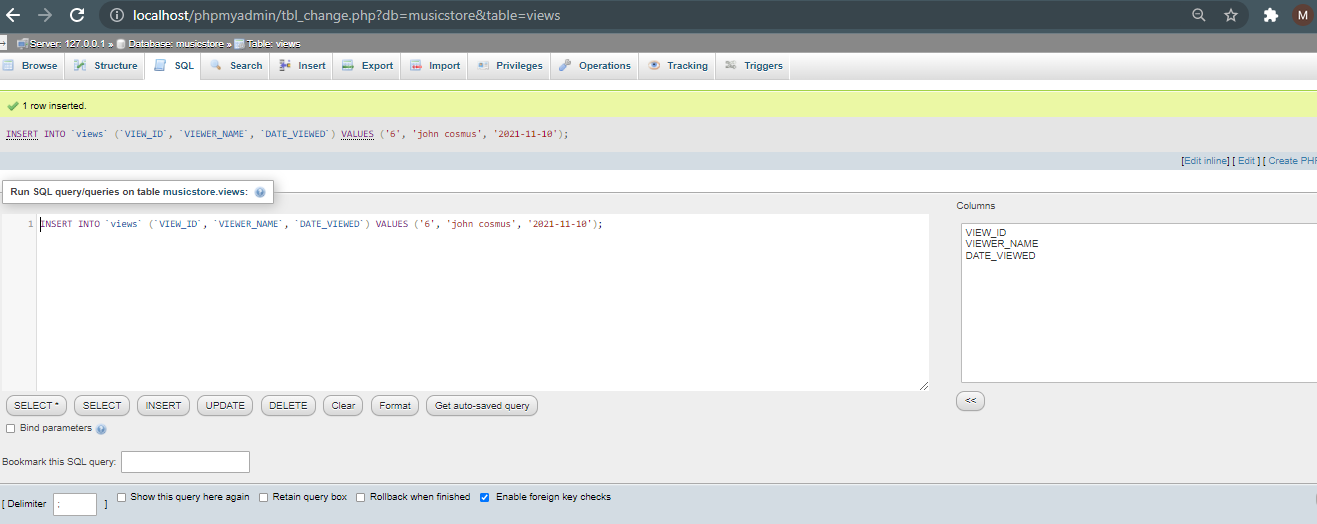
In table music\_store;



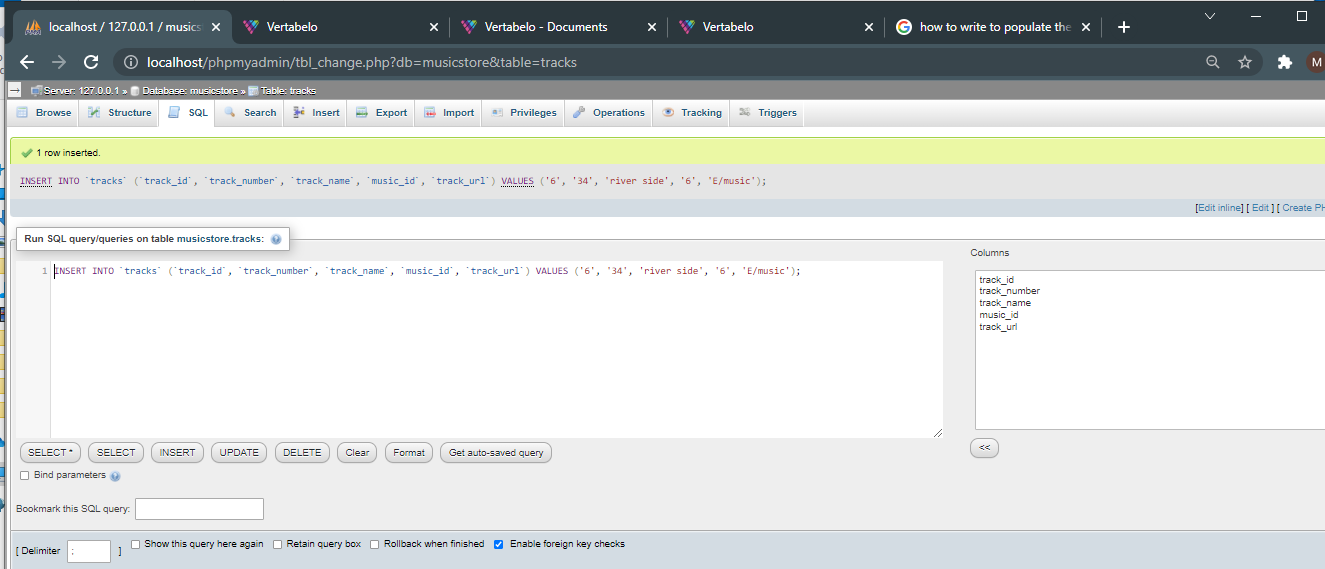
In table record\_label ;



In table view;

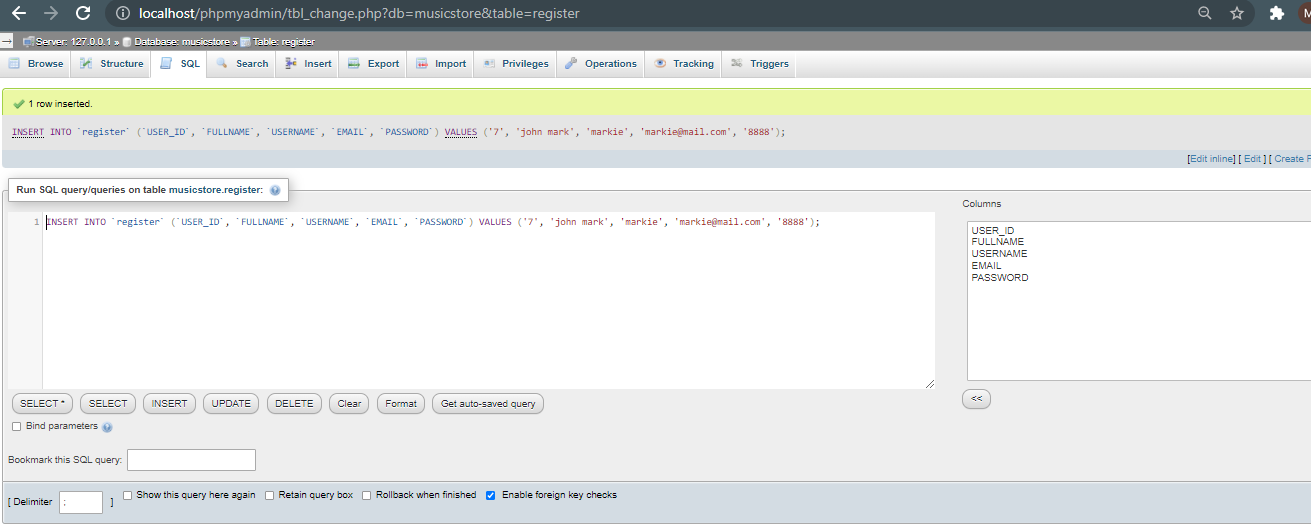


In table tracks ;



# 

# In table register;



# Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

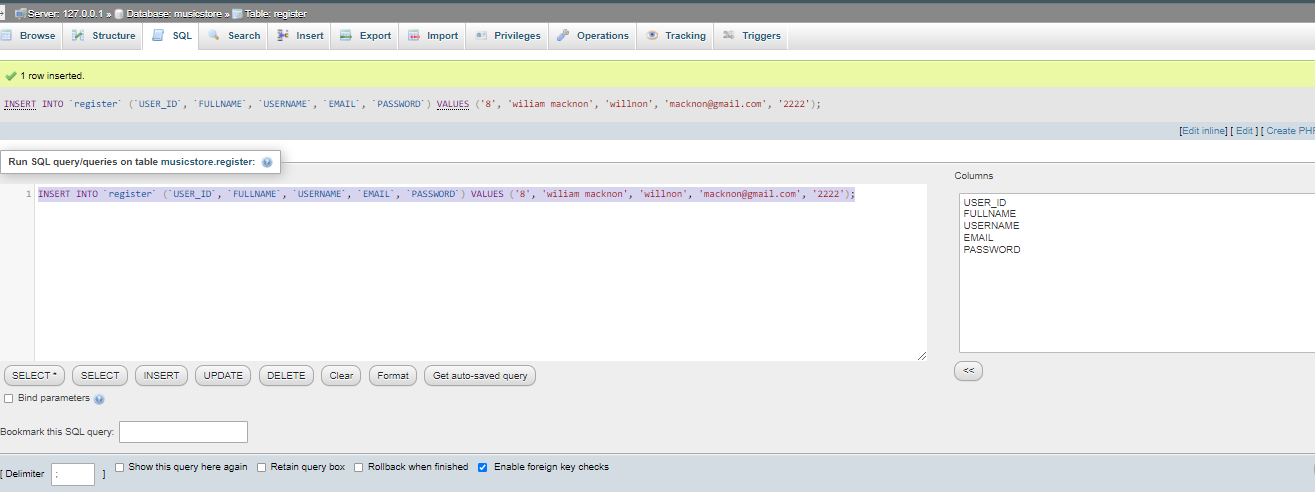
*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

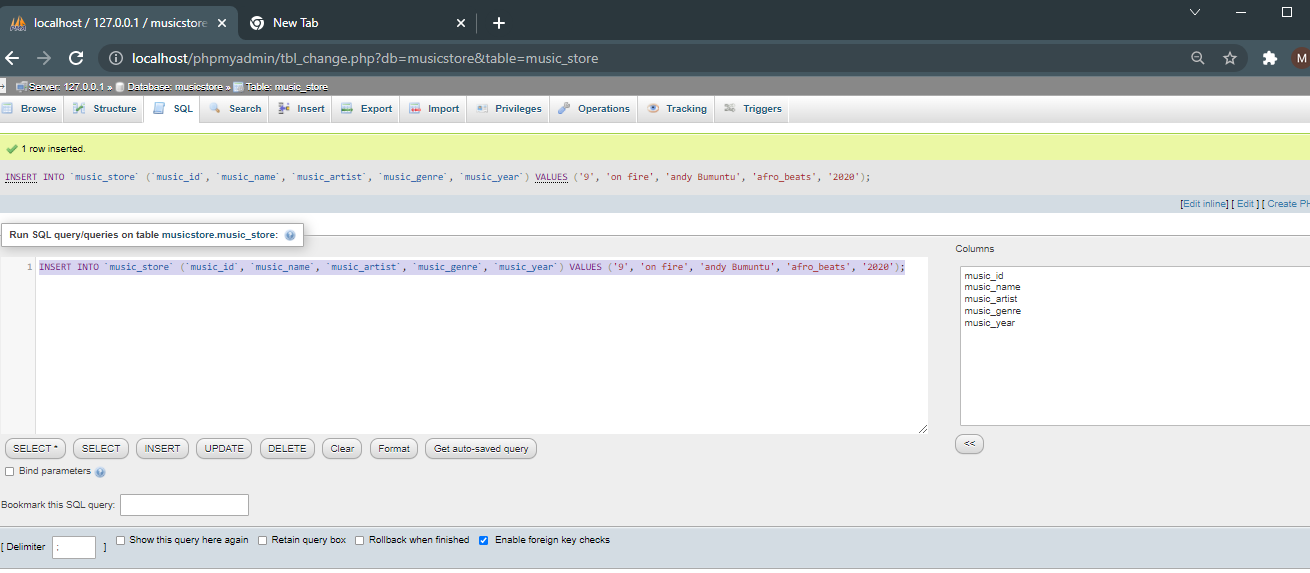
ENTER DML WORK HERE

INSERT INTO `register` (`USER\_ID`, `FULLNAME`, `USERNAME`, `EMAIL`, `PASSWORD`) VALUES ('8', 'wiliam macknon', 'willnon', 'macknon@gmail.com', '2222');



# 

# INSERT INTO `music\_store` (`music\_id`, `music\_name`, `music\_artist`, `music\_genre`, `music\_year`) VALUES ('9', 'on fire', 'andy Bumuntu', 'afro\_beats', '2020');



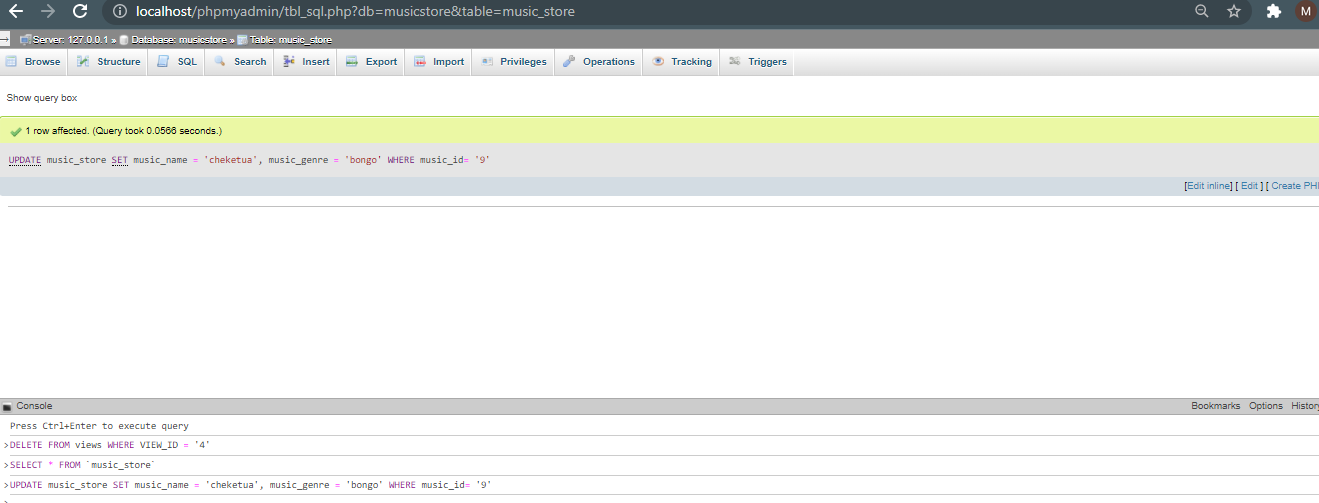
Update statements;

1. UPDATE music\_store

SET music\_name = 'cheketua',

music\_genre = 'bongo'

WHERE music\_id= '9';



# UPDATE record\_lebal

# SET LEBAL\_NAME = 'enb records',

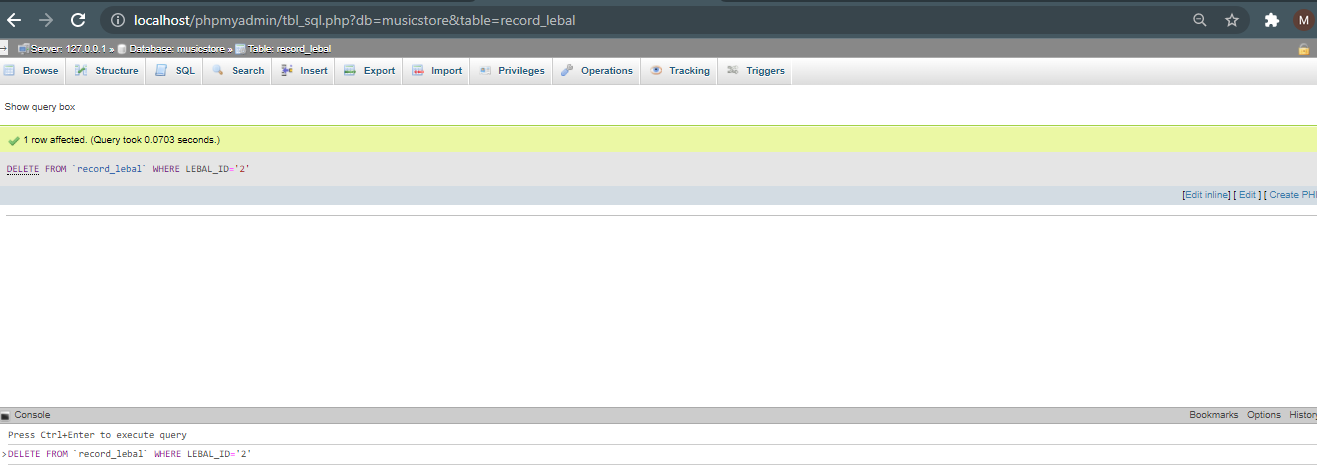
# music\_genre = 'raggae'

# WHERE lebal\_id= '4';

# Delete scrpts;

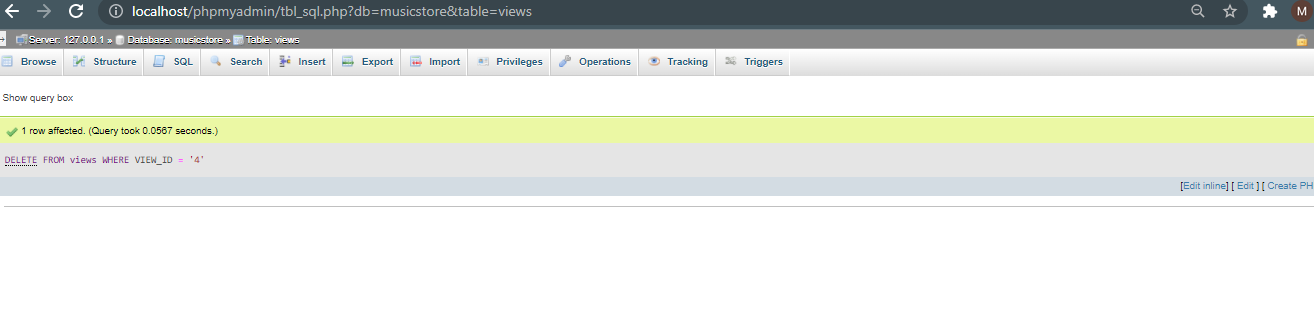
DELETE FROM record\_lebal

WHERE lebal\_id = ‘2’;



# DELETE FROM views

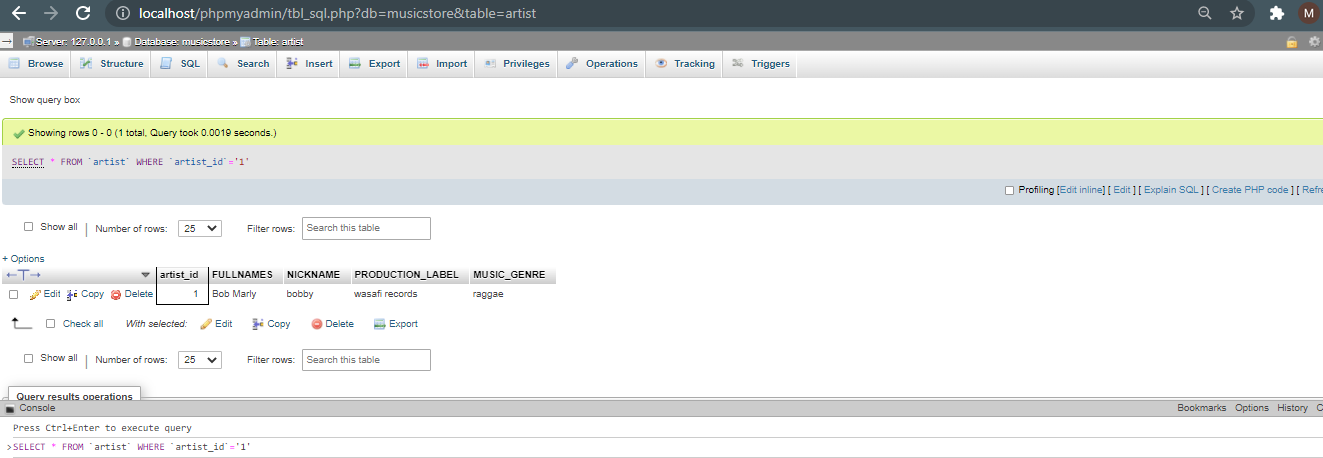
# WHERE VIEW\_ID = '4’;



Select statement;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `artist`

WHERE `artist\_id`='1’;



Join scripts;

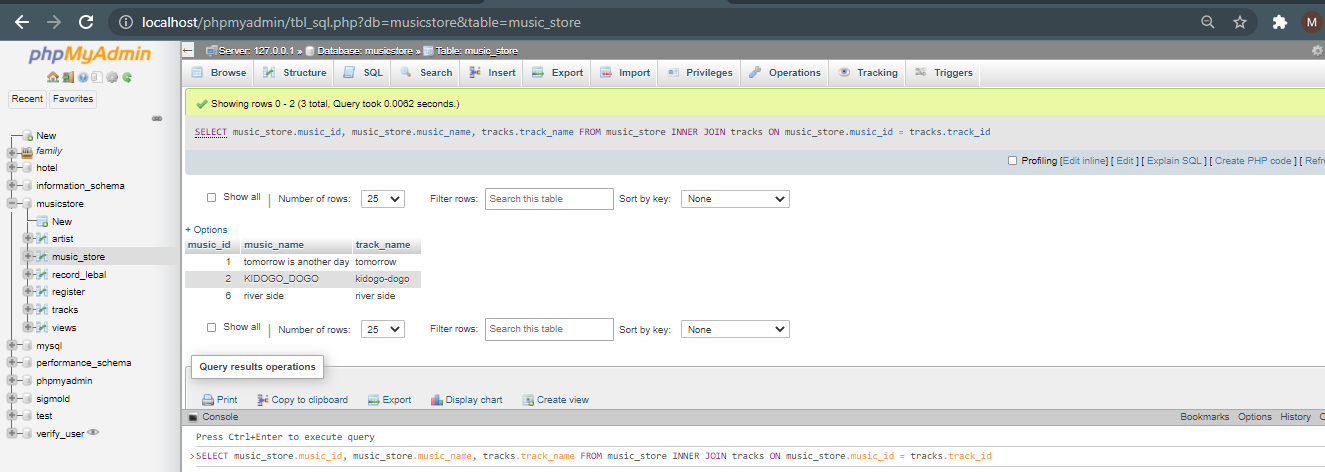
Inner join;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) music\_store.music\_id, music\_store.music\_name, tracks.track\_name

FROM music\_store

INNER JOIN tracks

 ON music\_store.music\_id = tracks.track\_id;



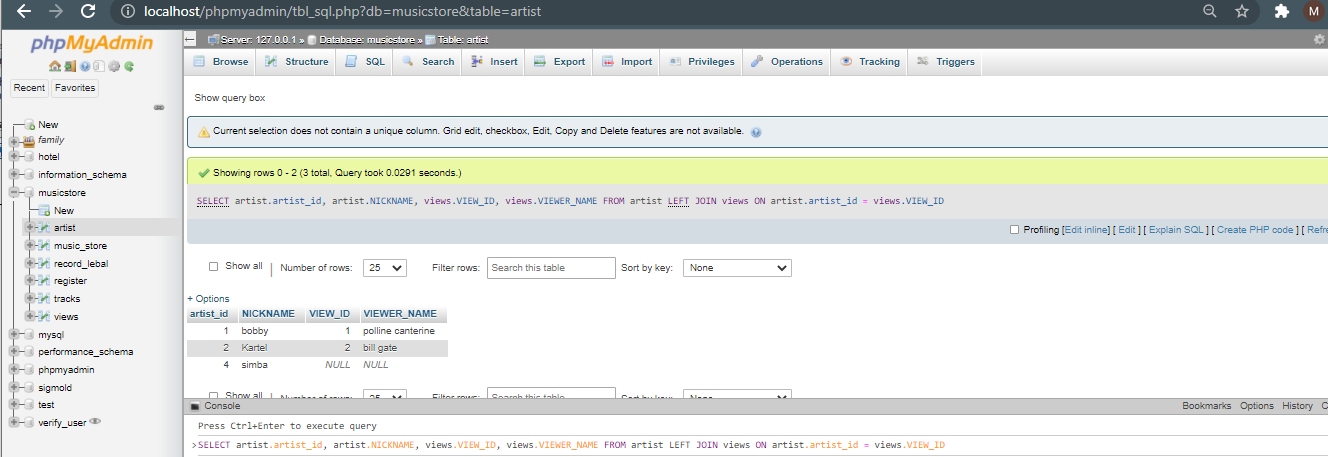
Outer join;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) artist.artist\_id, artist.NICKNAME, views.VIEW\_ID, views.VIEWER\_NAME

FROM artist

[LEFT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_left) JOIN views

ON artist.artist\_id = views.VIEW\_ID;



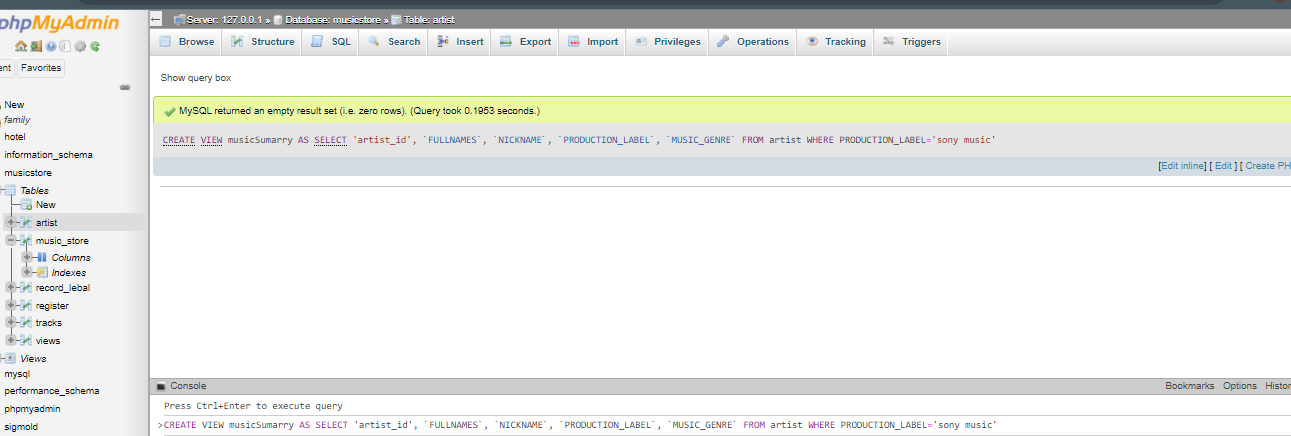
Summary statements;

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) [VIEW](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) musicSumarry

AS [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) 'artist\_id', `FULLNAMES`, `NICKNAME`, `PRODUCTION\_LABEL`, `MUSIC\_GENRE`

FROM artist

WHERE PRODUCTION\_LABEL='sony music';



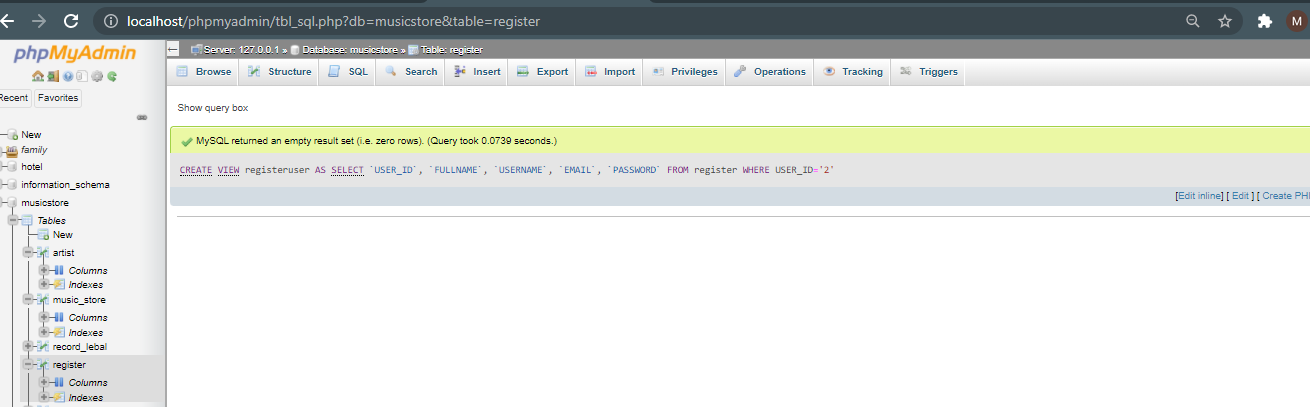
Second summary;

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) [VIEW](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) registeruser

 AS [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) `USER\_ID`, `FULLNAME`, `USERNAME`, `EMAIL`, `PASSWORD`

FROM register

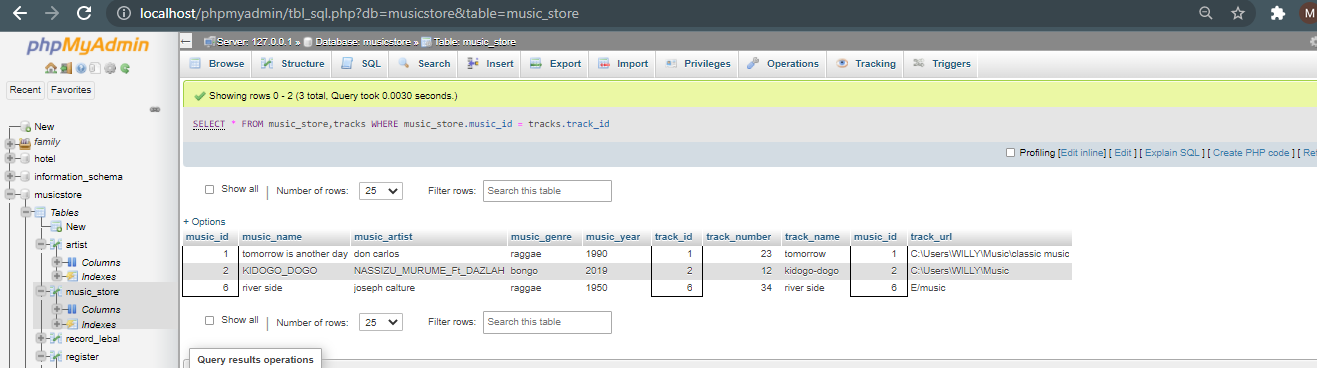
WHERE USER\_ID='2’;



Multiple table query;

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM music\_store,tracks

 WHERE music\_store.music\_id = tracks.track\_id;



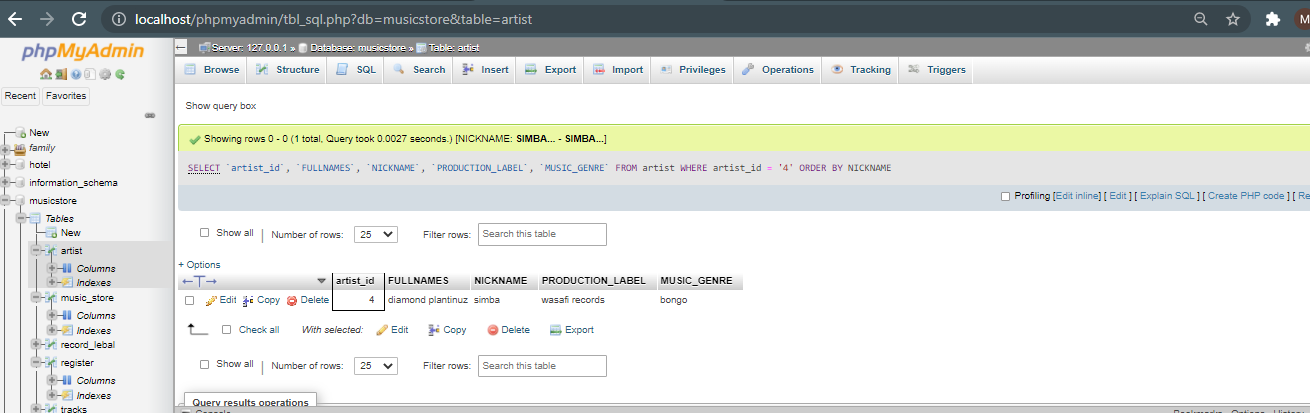
Query of choice;

SELECT `artist\_id`, `FULLNAMES`, `NICKNAME`, `PRODUCTION\_LABEL`, `MUSIC\_GENRE`

FROM artist

WHERE artist\_id = '4'

ORDER BY NICKNAME;



# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

ENTER YOUR INDEX WORK HERE

# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

ENTER YOUR WORK WITH VIEWS HERE

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRANSACTIONS HERE

# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 6*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

# PHP Programming

*Description: Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *4 points for writing an HTML form the user will use to enter search criteria*
* *8 points for a PHP script that uses the search criteria and returns results*
* *4 points for an HTML page that shows the results*
* *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
* *4 points for providing screen shots of your PHP website in action.*
* *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP DATABASE APP PROGRAMMING WORK HERE

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadv`antages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

# Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

MAKE AT LEAST THREE ENTRIES PER WEEK. CLEARLY IDENTIFY WHAT EACH PERSON ON YOUR TEAM ACCOMPLISHED. YOU MUST SHARE THE RESPONSIBILITY OF COMPLETING THE PROJECT.

|  |  |
| --- | --- |
| WEEK | ACTIVITY |
| Week one | * Second person: Title page * First person: Wrote the initial proposal to completion * Third person: Carried several research |
| Week two | * Third person: Ventured on data sources * First person: Worked on the alternatives of the relational database * Second person: set up the environment variables on the personal laptop to work on the project |
| Week three | * First person: dealt with various dependencies in the database tables * Second person: identified various entities * Third person: drew the database model in vertabello |
| Week four | * Second person: analyzed the boyce-codd rules explicitly * Third person: wrote the sql statements to populate the tables while taking screenshots * First person: wrote all the sql scripting statements and their respecting screenshots |
|  |  |
|  |  |
|  |  |